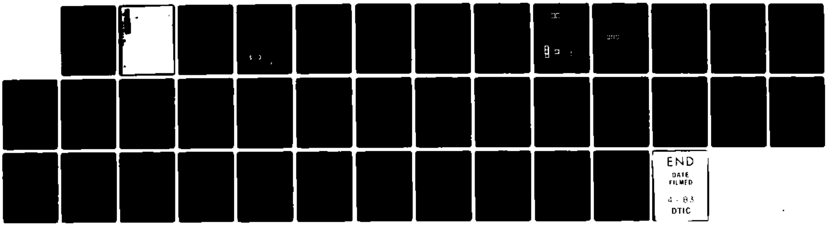
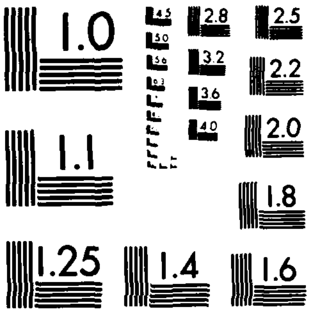
AD-A126 307 IMAGE PROCESSING AND DIGITIZATION FACILITY(U) HONEYWELL 1/1
SYSTEMS AND RESEARCH CENTER MINNEAPOLIS MN
K FANT ET AL. DEC 82 82SRC62 DAAK70-79-Q-0449

UNCLASSIFIED F/G 14/2 NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOV'T ACCESSION NUMBER AD-A126307	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (AND SUBTITLE) Image Processing and Digitization Facility		5. TYPE OF REPORT/PERIOD COVERED Final Report
		6. PERFORMING ORG. REPORT NUMBER 82SRC62
7. AUTHOR(S) Karl Fant and Scott Johnston		8. CONTRACT OR GRANT NUMBER(S) RFP #DAAK70-79-Q-0449
9. PERFORMING ORGANIZATIONS NAME/ADDRESS Honeywell Inc., Systems and Research Center 2600 Ridgway Parkway PO Box 312 Minneapolis, MN 55440		10. PROGRAM ELEMENT,PROJECT,TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME/ADDRESS U.S. Army MERADCOM Night Vision and Electro-Optics Laboratory Fort Belvoir, VA 22060		12. REPORT DATE December 1982
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME/ADDRESS (IF DIFFERENT FROM CONT. OFF.)		15. SECURITY CLASSIFICATION (OF THIS REPORT) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (OF THIS REPORT)

Distribution of this document is unlimited.

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (OF THE ABSTRACT ENTERED IN BLOCK 20, IF DIFFERENT FROM REPORT)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

Image Processing
Algorithms
Simulation

20. ABSTRACT (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

The Image Processing and Digitization Facility is an algorithm development and simulation laboratory developed by Honeywell Inc.

HD-168 REV 11/74

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 55 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

CONTENTS

LIST OF ILLUSTRATIONS

INTRODUCTION

The Image Processing and Digitization Facility (IPDF) is a sophisticated system of fully integrated hardware and software for engineering advanced image-processing algorithms.

The IPDF combines highly specialized hardware elements in a generalized software environment that provides consistent and convenient access to all elements of the system.

The IPDF provides the research scientist with command-level access to all image functions to facilitate quick evaluation and development of new algorithm ideas. The IPDF then supports the implementation of efficient simulations to test algorithms over large databases of images.

The IPDF is a research tool which provides all the facilities for design and development of state-of-the-art image-processing algorithms.

OVERVIEW

FACILITY DESIGN PHILOSOPHY

Image processing might be characterized as performing complex functions in relation to a simple data structure: the image. Because most image-processing revolves around this simple data structure, it is possible to provide specialized access techniques and device storage formats for the image data structure. This greatly simplifies function definition and maximizes computational efficiency. It is also possible to design specialized hardware which deals directly with the image data structure. Although images have a simple structure, they tend to be large and numerous, endowing image processing with a high computational cost. Because of this high computational cost and the possibility of a straightforward data management strategy for optimizing the computational environment, highly specialized systems become not only attractive but necessary.

The image-processing research environment demands a further dimension of system specification in terms of ease of access to image functions and the facility of combining image functions. These human interaction and internal system interaction issues must be carefully considered if a highly specialized image-processing system is to be flexible and a general image research tool. The system must:

- Generalize image access
- Maximize computational efficiency
- Provide easy access to image functions

LABORATORY HARDWARE CONFIGURATION

The IPDF consists of the Honeywell Level 6 Model 43 minicomputer, the I^2S (International Imaging System) Model 70F image display computer, an 875 line format video digitizer, a high-resolution color display, and a collection of standard video equipment (Figures 1, 2, and 3).

The Honeywell Level 6 Model 43 minicomputer provides the control and general purpose processing capability. It has one-half million bytes of central memory and one-half billion bytes of removable disk storage. It can handle large databases of digital images and efficiently implement Central Processing Unit (CPU) image functions.

The Level 6 uses the Mod 600 operating system, which is a multi-user, multi-tasking, time-sharing operating system based on the Multics time-sharing system. Mod 600 provides a sophisticated software development environment with convenient and powerful system extension facilities.

The I^2S Model 70F image display computer serves as the interface in both directions between the computer and the video realm. In addition, it provides rapid and powerful image-processing capabilities. As a display system, the Model 70 offers zoom, scroll, split screen, cursor, graphics overlay, and full-color output display from up to twelve 512 x 512 x 8 bit frame stores. It also provides real-time digitization of video images and real-time histograms of full images. As an image computer, it provides a feedback capability that processes a video image through three look-up table stages, each of which provides a general function capability and two arithmetic stages; it then stores the image in an image channel all in 1/30 of a second. This, combined with the scroll capability, provides a very fast capability for window-oriented convolution-type image functions.

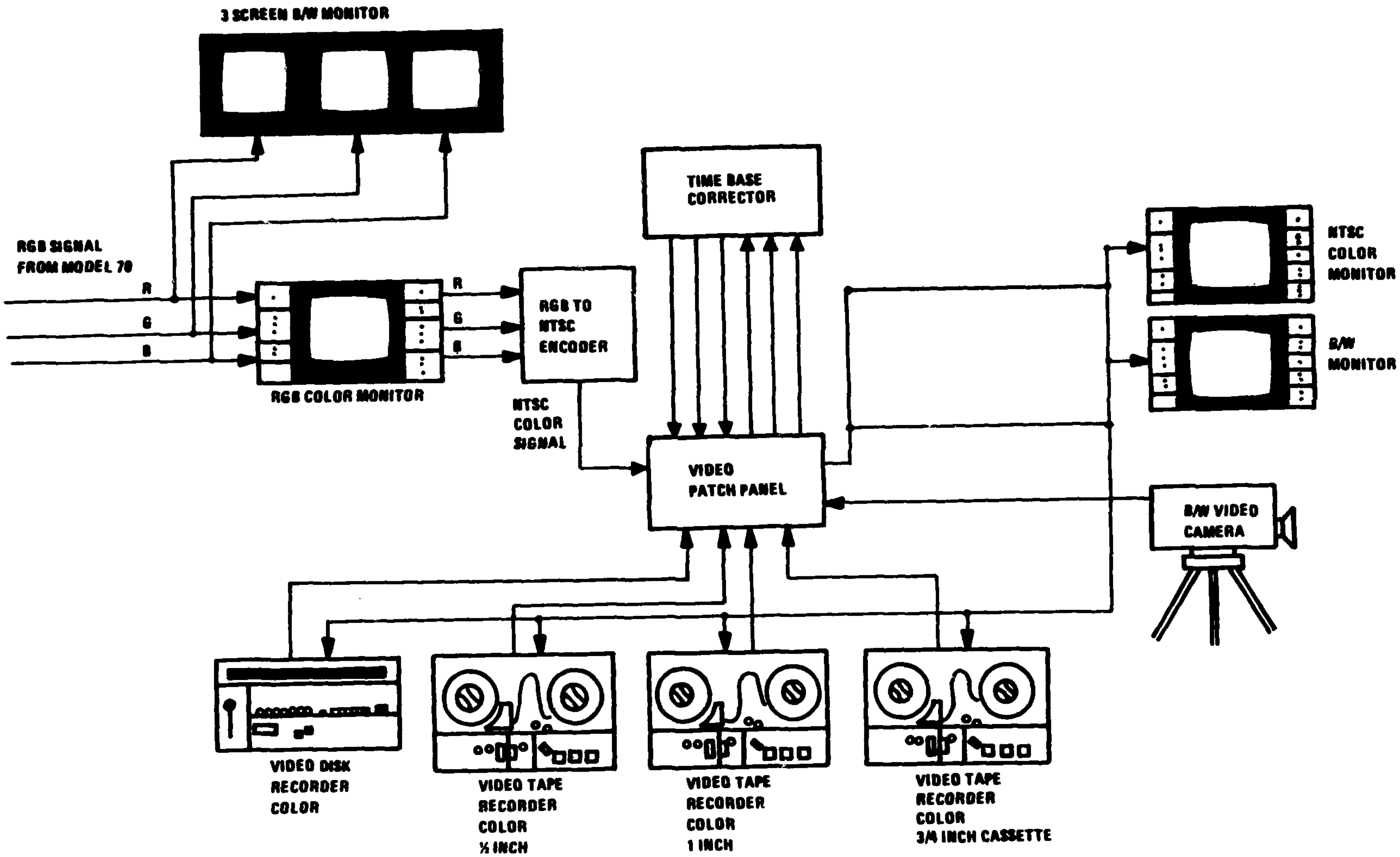


Figure 1. IPDF Video Components

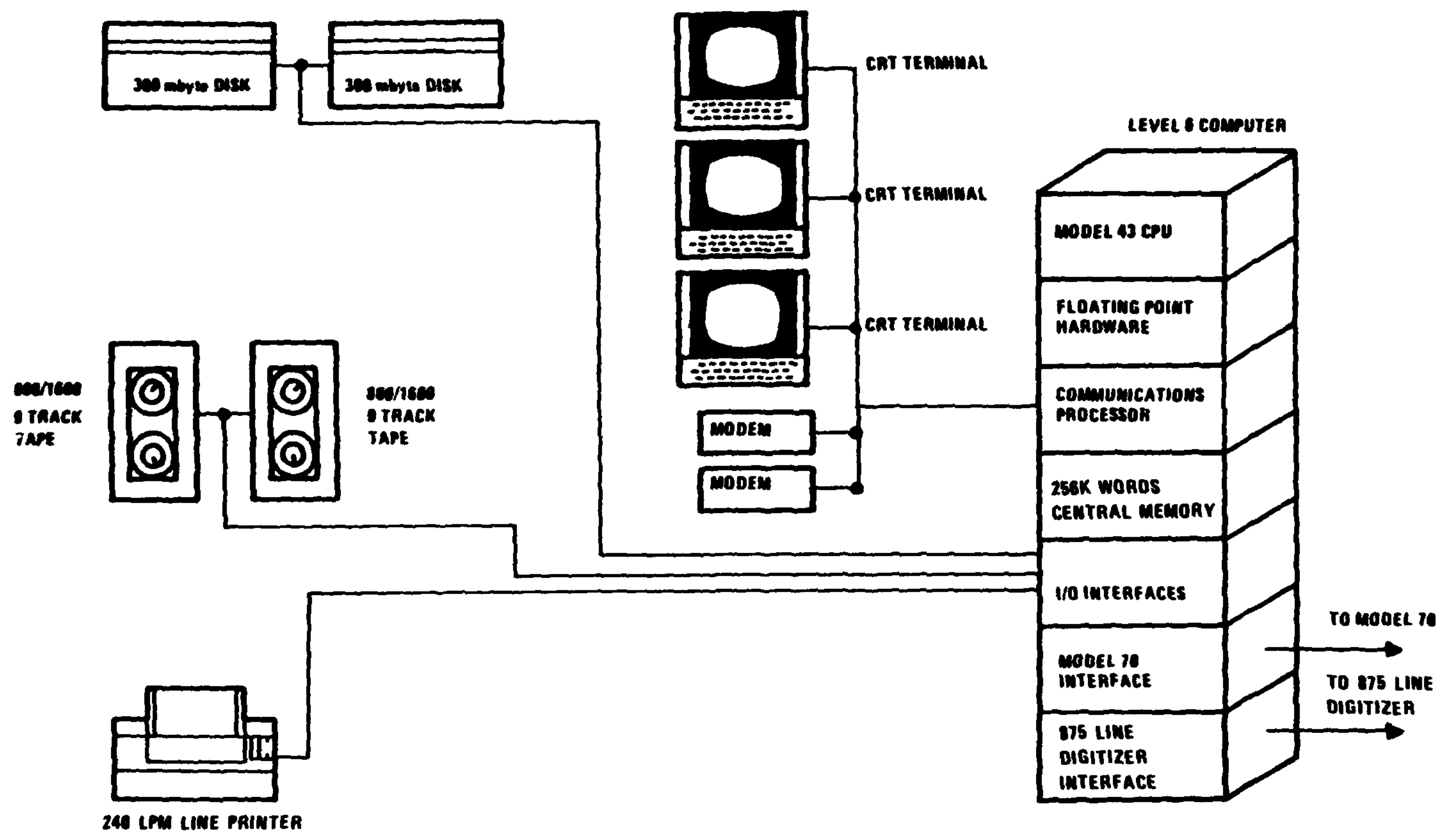


Figure 2. Facility Computer System

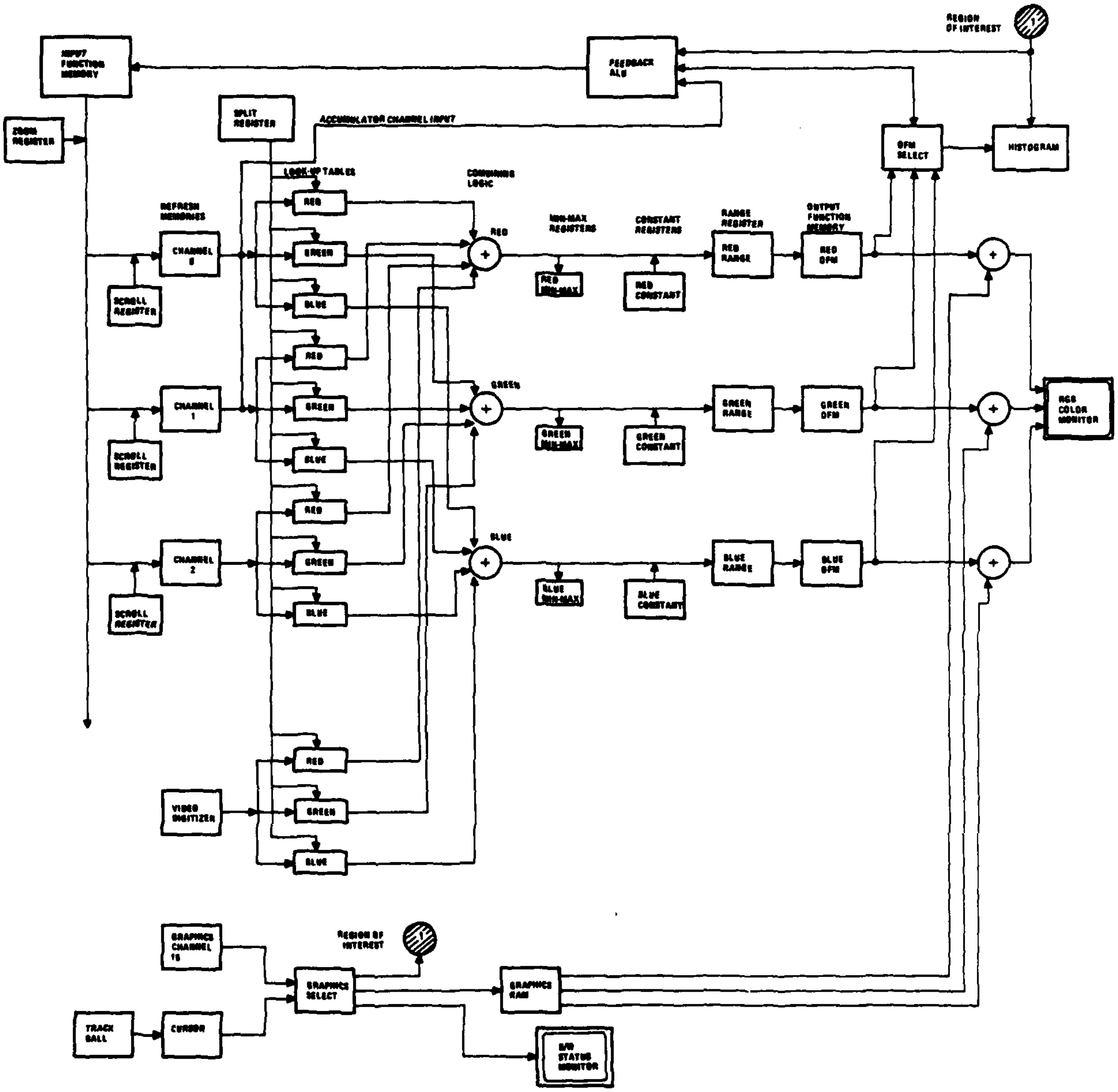


Figure 3. M70/F Architecture

An 875 line video format real-time digitizer provides access to 875 line format video data. The digitizer contains 1024 x 1024 x 8 bits of image storage that can be used by the image file system.

FACILITY SOFTWARE SYSTEM

The approach to software considers the image-processing facility as a group of interacting subenvironments. The design philosophy is to make each subenvironment a simple, straightforward, uniformly structured entity and provide simple, straightforward interfaces among the environments. Each subenvironment will be discussed individually. They are:

- Image environment
- Image function environment
- User environment, which consists of
  - Interactive environment
  - Development environment

Image Environment

The approach to the image environment (Figure 4) is based on the concept of the virtual image. Most image-processing systems provide a standard image file on a single device, typically disk or tape. Image data on other device types must be copied to an image file on the standard device type before being accessible to the image-processing system. The approach is to make image data on various storage devices in the system conform to a single virtual image format and access protocol so that an image function can access any image on any device in the system via a single-access protocol. The different devices are distinguished by naming conventions; once a file has been opened by name it is accessed via the virtual image protocol. Supported devices are disk, central memory, Model 70, and the 875 line digitizer.

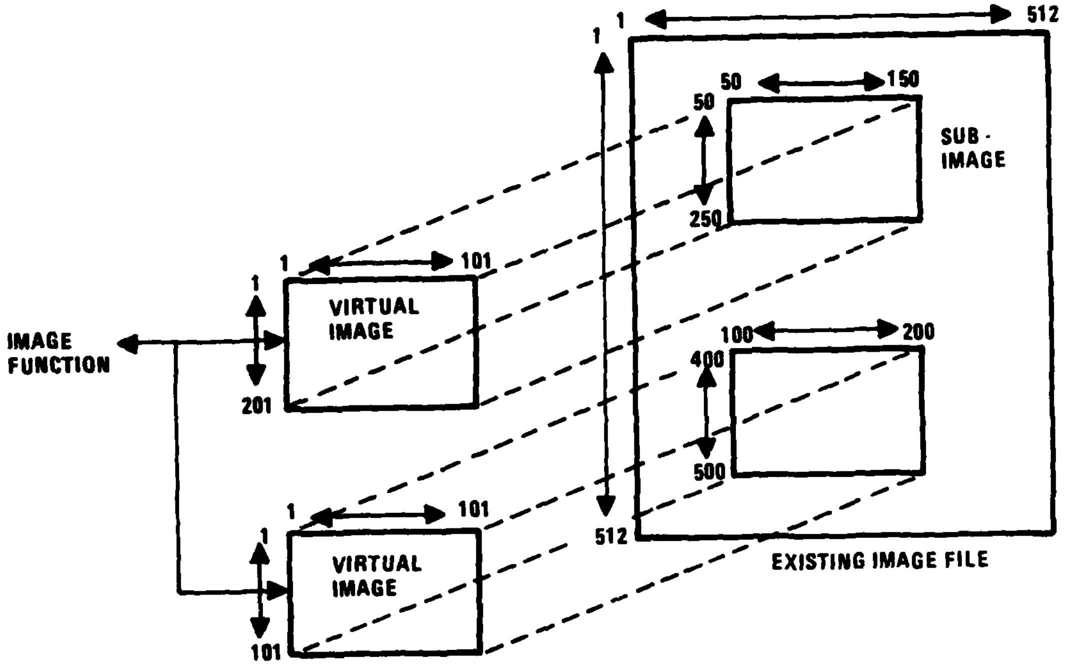


Figure 4. Subfile Capability

The virtual image is a very general image format. Many systems put limitations on the dimensions of the image files and offer a small set of information element formats. The facility image-file system allows the virtual image to be of any dimension within broad system constraints. The information can be any number of bits in size and may be signed integer data. The virtual image is accessed by specifying the image line and the portion of the line desired.

Another feature that greatly enhances the flexibility and generality of the facility image-file system is the capability to open any rectangular subportion of any existing image file as a fully autonomous virtual image; the system also has the ability to open several such subfiles of an existing image file simultaneously for all devices except central memory (Figure 4). Once opened, the subfile is accessed via the virtual image just like any other file in the system.

In summary, the image environment provides:

- Uniform access convention to image data independent of the storage device the data resides on
- An identical access convention to any subportion of an existing image

Image-Function Environment

The image-function environment (Figure 5) consists of routines that perform various functions on or in relation to an image. Operating on images is their only general common feature, but the availability of a single-image format and access protocol allows them to share a common internal structure as well as a common communication with the user and communication with the image.

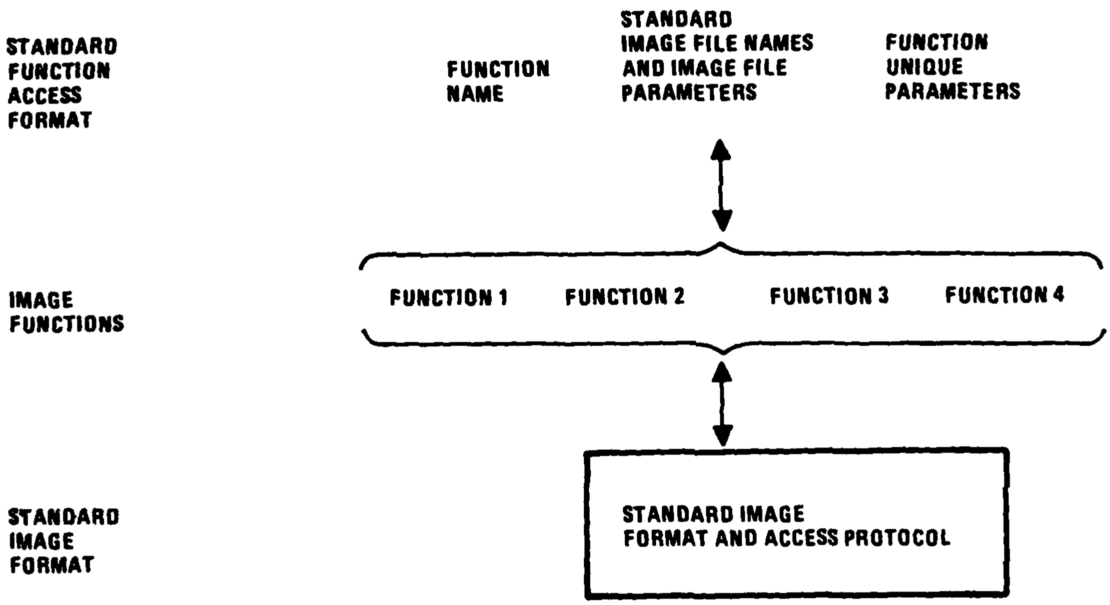


Figure 5. Image-Function Environment

An image-file system provides not only a uniform file format and access protocol but also provides a standard image-file identification and parameter specification protocol. Thus, function routines can utilize a standard interface for receiving their operating parameters as well as a standard interface for performing their functions. The result is a highly structured function environment which assists and guides the development of function routines and allows totally general function routines that can perform their function on any image file in the system via a simple and straightforward requesting procedure.

The key to a consistent and general function environment is a flexible and comprehensive image-file system. The function environment still has to be carefully designed and managed. It doesn't automatically follow from an image-file system even though the file system is a necessary prerequisite and a big help.

In summary, the image-function environment considerations are:

- Uniformity of interface to image
- Commonality of internal structure maintained as completely as possible
- Standardized access conventions to the image functions

User Environment

The image-processing user is, of course, the primary reason for the existence of the system, and any design must provide for fast, convenient, and efficient service to the user. The user environment (Figure 6) consists of two subenvironments in the interactive environment and the development environment.

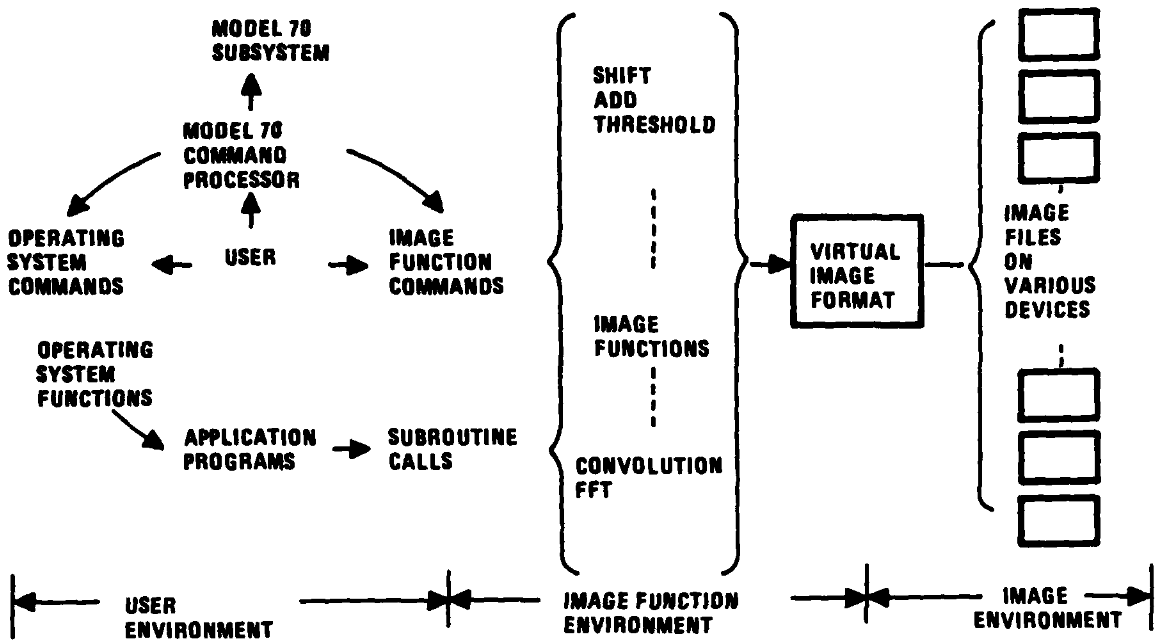


Figure 6. System Environment

Interactive Environment--The interactive environment consists of:

- Operating system commands and subsystems
- Image function commands
- Model 70 control command processor

The facility approach is to base the environment on a flexible operating system and to implement the image-processing system as a fully integrated extension of the host operating system. The Honeywell Level 6 Mod 600 operating system provides a firm foundation for extension and integration.

Specialized commands can be easily created and installed with full access to the system command processor facility. As such, they become fully integrated aspects of the general system environment. The further strategy of utilizing the Mod 600 command format convention for the image commands maintains consistency in the interactive environment.

The Model 70 control command processor is a special subsystem to access the Model 70 image display computer. It is implemented as a fully integrated subsystem of Mod 600. This means that all the system capabilities, including special image function commands, are accessible from the Model 70 control command processor. This creates an interactive environment with a uniform and consistent structure and format through which all capabilities of the system, both standard and operating system and specialized image processing, are conveniently available.

In summary, the interactive environment considerations are:

- A flexible operating system
- Modeling the image-processing command environment after the operating system command environment so that the specialized image-processing functions appear in all respects as integral system functions

Development Environment--The development environment supports the generation of new functionality from existing functionality. The facility approach is to support this capability at both the command level and the program level. Development is supported at the program level by maintaining callable subroutines corresponding to each image-processing command. Under the Mod 600 operating system there is a simple technique which converts any subroutine into a system command. This is precisely how image function commands are implemented. The command-subroutine correspondence can be easily maintained.

Development at the command level is supported by providing command file interpretation whereby several commands can be combined in a sequential order to carry out a higher level image function.

The Mod 600 operating system has a sophisticated command file interpretation facility. All image functions implemented as system commands can be acessed via a command file. The Model 70 command processor implemented as an operating system subsystem can be accessed via a command file and passes Model 70 commands from successive lines of the command file. The complete specialized image-function capability is accessible from system command files.

The Model 70 command processor also has a command file interpretation facility. It is capable of nested command files, parameterized command files, and accessing the full capability of the operating system. Any operating system command can be accessed, including requesting the operating system to interpret an operating system command file.

This fully integrated command file capability provides a powerful facility for combining functionality at the command level. Image-function experiments can be quickly and easily set up via commands as a preliminary test of the function itself or as a means of debugging the implementation of the function.

After the interactive phase, the command files can be translated directly into subroutine calls in a program for more efficient operation during final design verification on large databases.

In summary, the development environment considerations are:

- Maintain full correspondence between command-level capabilities and subroutine callable capabilities
- Provide a sophisticated command file interpretation facility for combining functions at the command level
- Maintain full accessibility of any specialized subsystem command files to the capabilities of the operating system and full accessibility of the operating system command files to the capabilities of any subsystem

Software Management and Maintenance

All software installed in the system must conform to interface standards and internal documentation standards and must be reviewed by two reviewers. This ensures that the system evolves consistently within the established design principles.

The source for all installed routines is maintained in a special on-line directory. The internal documentation is available on line via the - HELP parameter or INFO command. An on-line catalog that contains a one-line description of all installed routines or commands is also maintained.

Algorithm Development on the Facility

The first stage of algorithm development is to digitize and store on disk a small number of frames of the imagery in question. These images can then be used for preliminary interactive command-level processing functions. Intermediate result images can be stored and submitted to further processing.

In this manner, image-processing algorithms can be developed from a library of basic functions. At each stage of the development, immediate visual display of intermediate results is available on which to base an engineering judgment on the effectiveness of the algorithms. If the results do not look promising, other functions can be tried or parameters can be adjusted. New functions can be implemented and become part of the standard library. This process is continued to gradually build a file of command sequences that represent an approximation of the desired algorithms. Once the engineer is satisfied with the interactive command-level study, the command sequence can be translated into a more efficient simulation program by translating the commands into subroutine calls. Appropriate functions can also be identified for implementation in the Model 70. The Model 70 provides extremely fast execution of certain classes of image functions. The resulting simulation can then be used to process a large set of imagery to test the algorithm more thoroughly.

APPENDIX

SOFTWARE COMPONENTS

IPDF CATALOG OF SUPPORTED SUBROUTINES

12/01/82

```
*
*   CPU IMAGE FUNCTIONS
*
*
CSOBEL     Cpu SOBEL gradient operator
CAVG       Cpu window AVeraGe
FCMEDN     Fast Cpu MEDiaN filter
CCMPAS     Cpu CoMPASs gradient operator
CRBRTS     Cpu RoBeRTS gradient operator
CNSCLN     Cpu NoiSe CLeaNing function
CKIRSH     Cpu KIRScH edge enhancement
CMASK      Cpu general convolution (MASK) function
CLAPLN     Cpu LAPLaciaN operator
CWINTH     Cpu WINdow average THresholding
CCORR      Cpu CORRelation of two images
CONCMP     CONnected CoMPonent analysis
HISTEQ     HISTogram EQualization
RANDIM     RANDom noise IMage generation
CLAGBC     Cpu Local Area Gain and Brightness Control
CHSTHP     Cpu HiSTogram HyPerbolization
CHSTEQ     Cpu HiSTogram EQualization
CPTRAN     Cpu PoinT TRANsformation
CSIZE      Cpu continuous SIZEing (shrink or expand)
C2DFHT     Cpu 2 Dimensional Fast Hadamard Transform
*
*   CPU SUPPORT ROUTINES
*
*
CPFILE     CoPy image FILEs (or feature vector files)
IMGMSE     IMaGe Mean-Squared-Error & snr calculation
IMDUMP     IMage DUMP to user_out
CHIST      Cpu image HISTogram
CHISTM     Cpu image HISTogram after Mapping
GLDTM      Gray Level Dependency Texture Measure
SGLDTM     Spatial Gray Level Difference Texture Measure
```

```
*
*   MODEL 70 IMAGE FUNCTIONS
*
*
M7MASK     M70 general convolution (MASK) function
MAVG       M70 window AVeraGe
FMAVG      Fast M70 window AVeraGe
MABSAC     M70 ABSolute value of ACcumulator
MSOBEL     M70 SOBEL gradient operator
MRBRTS     M70 RoBeRTS gradient operator
MNSCLN     M70 NoiSe CLeaNing function
MACDIV     M70 ACcumulator DIVision
MLAPLN     M70 LAPLaciaN operator
MCMPAS     M70 CoMPasS gradient operator
MSCACC     M70 SCale ACCumulator
MROOTA     M70 sqare ROOT of Accumulator
M70MLT     M70 MuLTiplication
MHSTHP     M70 HiSTogram HyPerbolization
MHSTEQ     M70 HISTogram EQualization
MWINMX     M70 WINdow MaXimum
MWINMN     M70 WINdow MiNimum
M2DFFT     M70 2 Dimensional Fast Fourier Transform
M2DFHT     M70 2 Dimensional Fast Hadamard Transform
MMEDAL     M70 MEDiAL axis operator
*
*   MODEL 70 SUPPORT ROUTINES
*
*
M7INIT     M70 INITialize
M7PERP     M70 PERPendicular image file opener
PSEUDO     PSEUDO color generation
M7ANOT     M70 ANnOTate characters
M7CHAR     M70 annotate CHARacters (improved)
TBSUB      TrackBall SUBimage definition (one relative)
TBPOS      TrackBall POSition (one relative)
CURSUB     CURsor SUBimage definition (zero relative)
CURPOS     CURsor POSition (zero relative)
TBWAIT     TrackBall WAITing for button push or cursor motion
TBINT      TrackBall waiting (INT) (old version)
ONECHN     check mask for ONE CHaNnel
MCLEAR     M70 CLEAR selected channels and planes
AXIS       draw and annotate AXIS
DRAWLN     DRAW LinE
DRAWPT     DRAW PoinT
DRAWBR     DRAW BaR
M7VECT     M70 draw VECTor
```

```
*
*   FEATURE VECTOR FUNCTIONS
*
*
--none yet--
*
*   FEATURE VECTOR SUPPORT ROUTINES
*
*
MEANS      compute feature vector MEANS
CLINFO     CLass INFOrmation
```

```
*
*   IMAGE FILE SUBSYSTEM
*
*
OPENIM      OPEN IMage file
CLOSIM      CLOSe IMage file
IMINFO      get IMage file INFOrmation
IMPATH      get IMage file PATHname
IMXIST      test IMage file eXISTence
IMOFFS      get subIMage OFFsets
RDLINE      ReaD image LINE
WRLINE      WRite image LINE
RDCOL       ReaD image COLumn
WRCOL       WRite image COLumn
RDHEAD      ReaD image file HEADer string
WRHEAD      WRite image file HEADer string
*
*   FEATURE VECTOR FILE SUBSYSTEM
*
*
OPENFV      OPEN Feature Vector file
CLOSFV      CLOSe Feature Vector file
FVINFO      get Feature Vector file INFOrmation
FVPATH      get Feature Vector file PATHname
FVXIST      test Feature Vector file eXISTence
FVEOF       Feature Vector file End-Of-File
FVXPND      Feature Vector file eXPaND
RDVECT      ReaD Feature Vector
WRVECT      WRite Feature Vector
RDFEAT      ReaD selected FEATures from vector
WRFEAT      WRite selected FEATures to vector
*
*   MAGTAPE HANDLING SUBSYSTEM
*
*
OPNTAP      OPeN TAPe device
RDTAP       ReaD TAPe record
WRTAP       WRite TAPe record
CLSTAP      CLoSe TAPe device
SKRTAP      SKip Records on TAPe
SKFTAP      SKip Files on TAPe
EOFTAP      write End-Of-File TAPe mark
REWTAP      REWind TAPe
UNLTAP      UNLoad TAPe
*
*   SLIDING WINDOW SUBSYSTEM
*
SLWNIT      SLiding WiNdow IniTialization
SLWNLN      SLiding Window Next LiNe
SLWNT1      SLiding WiNdow IniTialization (no output image)
SLWNL1      SLiding Window Next LiNe (no output line)
```

```
*
*   CLASSIFIER SUBSYSTEM
*
*
PREPROCESSORS
*
COVAR      compute COVARiance matrices
INVCOV     INVert COVariance matrices
DIVERG     compute DIVERGence measure
KLEXP      Karhounen-Lowe EXPansion calculation
KLINFO     Karhounen-Lowe expansion INFOrmation
KLTRAN     Karhounen-Lowe expansion TRANsformation
*
TRAINERS
*
KNNTRN     K-Nearest-Neighbor classifier TRaiNing
KNNINF     K-Nearest-Neighbor classifier training INFormation
BAYTRN     BAYesian classifier TRaiNing
BAYINF     BAYesian classifier training INFormation
*
CLASSIFIER
*
KNNCLS     single sample K-Nearest-Neighbor CLaSsifier
BAYCLS     single sample BAYesian CLaSsifier
*
CONFUSION ANALYSIS
*
CONFU      CONFUsion matrix manager
KNNCON     K-Nearest-Neighbor classifier with CONfusion matrix
BAYCON     BAYesian classifier with CONfusion matrix
```

```
*
*   NUMERIC FUNCTIONS
*
*
UNIRAN     UNIformly distributed RANdom number generation
GAURAN     GAUssian distributed RANdom number generation
EXPRAN     negative EXPonentially distributed RANdom number generation
POIRAN     POIsson distributed RANdom number generation
SYMEIG     real SYMmetric matrix EIGen vectors and values
MATMLT     real MATrix MuLTiplication
MATADD     real MATrix ADDition
TAYSIN     TAYlor series SINe
TAYCOS     TAYlor series COSine
POWR2      nth POWeR of 2 (used to set bits)
SETBUF     SET integer BUFfer with linear function
FFT        Fast Fourier Transform
FHT        Fast Hadamard Transform
BCKTRK     BaCKTRacKing for fht
SCITOI     SCale Integer array TO Integer array
SCRTOI     SCale Real array TO Integer array
IMNMX      Integer array MiN and MaX
RMNMX      Real array MiN and MaX
*
*   OPERATING SYSTEM ROUTINES
*
*
CRFIL      CReate sequential FILe
RLFIL      ReLease FILe
RMFIL      ReMove FILe
RNFIL      ReName FILe
GTFIL      GeT FILe
GTFILE     Routine to GeT and open a sequential FILE
RMFILE     Routine to ReMove a FILE
XPATH      eXpand file PATHname
TRMRQ      TeRMinate  ReQuest (task) with error code
CMDLN      execute system CoMmanD LiNe
CHAIN      load and execute overlay (CHAINing)
CRASH      CRASH task and activate the dump utility
AFNTST     Active FuNction mode TeST
AFNRET     Active FuNction character string RETurn
GMEMA      Get dynamic MEMory (Available memory only)
RMEM       Return dynamic MEMory
*
*   I/O ROUTINES
*
*
CIN        read directly from Command_IN file
USIN       read directly from USer_IN file
USOUT      write directly to USer_OUT file
USOUTF     write to USer_OUT with Fortran carriage control
EROUT      write directly to ERror_OUT file
EROUTF     write to ERror_OUT with Fortran carriage control
RPTER      RePorT system defined ERror
INTIN      input INTEGER value from user_IN
GETINT     GET INTeger value from user_in
REALIN     input REAL value from user_IN
HEXIN      input HEXadecimal value from user_IN
HEXDMP     HEXadecimal DuMP of integer data
```

STRDMP	STRing DuMP to user out
WRHDR	WRite a sequential file HeaDeR
RDHDR	ReaD a sequential file HeaDeR
CKHDR	ChecK a sequential file HeaDeR
USRID	get a USeR ID
$WR	assembly language text WRiter declaration macro
$WRNUM	assembly language WRite NUMbers macro
$WRSTR	assembly language WRite STRings macro
$WRADD	assembly language WRite ADDress macro

```
*
*   STRING MANIPULATION ROUTINES
*
*
PACK       generalized bit string PACKing
UNPACK     generalized bit string UNPACKing
PAKB       PACK Bytes
UNPAKB     UNPACK Bytes
PBITS      Pack BIT value array to Single word
GBITS      Get BIT value array from Single word
CTOI       move Character variable data TO Integer variable
ITOC       move Integer variable data TO Character variable
IVAL       return Integer VALue of character
FMTSCN     ForMaT SCaN character string for decimals, digits and blanks
DEBLNK     DElete BLaNKs from character string
UNBLNK     UN (delete) BLaNK a character string
APPEND     APPEND character strings
STRCPY     character STRing CoPY
*
*   COMMAND LINE PARSING ROUTINES
*
*
RTNPAR     ReTurN PARameters from command line
PNORIN     PathName accessing OR string of INtegers parsing
PNORRL     PathName accessing OR string of ReaLs parsing
CHNCOL     parse string for m70 CHaNnel and COLor specification
CTOINT     convert Character string of decimal digits to INTeger value
CTORL      convert Character string of decimal digits to ReaL value
NEXTCH     return NEXT CHaracter string from character array
*
*   DEBUG ROUTINES
*
*
WRFLAG     WRite FLAG
FLAG       read FLAG
FLGIN      FLaG INput from disk file
```

```
*
*   MODEL 70 INTERFACE ROUTINES
*
*
ALU       read/write ALU control registers
CONST     read/write CONSTant registers
CRCTL     read/write CursoR ConTroL register
CURSR     read/write CURSoR position register
FDBCK     write FeeDBaCK control and initiate feedback
GRAFE     read/write GRAphics control register
GRRAM     read/write GRaphics color assignment RAM
IFM       read/write Input Function Memory
IMAGE     read/write IMAGE refresh channels
LUT       read/write Look Up Tables
LTCNT     write Look up Table CoNnecTion
MNMAX     read MiNMAX registers
OFM       read/write Output Function Memories
RHIST     Read HISTogram tables
SCROL     read/write SCROLl registers
SHIFT     read/write SHIFT registers
SPLIT     read/write SPLIT screen tables
STCUR     read/write (SeT) CURsor shape memory
ZOOM      write ZOOM control registers
M70VR     return M70 VeRsion
M70OP     return M70 OPtions
MKHDR     MaKe HeaDeR for M70XF transmission
M70XF     M70 transfer (XF) routine
```

```
*
*   MODEL 70 PRIMITIVES
*
*
ABORT      iis ABORT routine
BCHAN      Blank image CHANnel
CSCLR      m70/CS CoLoR determiner
DADRS      convert channel number to channel mask
DCURS      Display new CURSor shape
DEXEC      dummy routine for HP3000 compatibility
DHIST      compute HISTogram
DMASK      function to convert channel number to channel mask
DPLUS      write PLUS on image channel
DWAIT      iis WAIT routine
EXOFM      load EXponential table in OFMs
GROFF      dummy call to OFGRF
HCLIP      Histogram based CLIPping
HSTYP      select HiSTogram TYPe (color)
IMOD       iis MOD function
I4         iis function to convert Integer*4 to Integer*2
INOT       iis NOT function
IXOR       iis IEOR function
LGLUT      load LoGarithmic table in LUTs
LNIFM      load LiNear ramp in IFM
LNLUT      load LiNear ramp in LUTs
LNOFM      load LiNear ramp in OFMs
OFGRF      selectively turn OFf GRaph(F)ics planes
OFMLD      LoaD OFM with linear ramp with specified max
ONCUR      turn ON CURsor
PROFL      iis PROFiLe control register
RBUTN      Read BUTtoNs and cursor position
SETUP      iis SETUP routine
STCOL      SeT COLor of graphics planes
TKHIS      TaKe HIStogram via the videometer
VIDEO      iis read/write video control info
WAITB      WAIT for Button push
XCOLR      set graphics intersection (X) COLoRs
XLATE      trans(X)LATE color mask to number
ZBUFF      Zero out BUFFer
ZOOMC      compute actual ZOOMed Coordinates
```

```
*
*   SOFTWARE PACKAGE SAVE FILES
*
*
SIG_PROC   IEEE digital SIGnal PROCessing package
STATS      STATiStical package
```

```
**********************************************************************
**********************************************************************
*                                                                    *
*   1PDF CATALOG OF SUPPORTED COMMANDS                               *
*                        11/23/82                                    *
*                                                                    *
**********************************************************************
**********************************************************************
```

```
*
*   CPU IMAGE FUNCTION COMMANDS
*
*
CSØBEL         Cpu SOBEL gradient operator
CAVERAGE       Cpu window AVERAGE
FCMEDIAN       Fast Cpu MEDIAN filter
CCOMPASS       Cpu COMPASS gradient operator
CROBERTS       Cpu ROBERTS gradient operator
CNOISECLN      Cpu NOISE CLeaNing function
CKIRSCH        Cpu KIRSCH edge enhancement
CWINTHRSH      Cpu WINdow THReSHold operation
CLAPLACN       Cpu LAPLACiaN operator
CWINMASK       Cpu WINdow MASKing general convolution operator
CCROSSCOR      Cpu CROSS CORrelation
CONNCOMP       CONNected COMPonent analysis
RANDIMAGE      add RANDom noise to an IMAGE
CLAGBC         Cpu Local Area Gain and Brightness Control
CHISTHYP       Cpu HISTogram HYPerbolization
CHISTEQU       Cpu HISTogram EQUalization
CONSIZE        Cpu cONtinous SIZEing (shrink or expand)
CP2DFHT        CPu 2 Dimensional Fast Hadamard Transform
*
*   CPU SUPPORT COMMANDS
*
*
CREATIM        CREATe IMage file
IMINFO         IMage file INFOrmation
TEXTRDIFF      TEXTuRe DIFFerence measure
TEXTRDEPN      TEXTuRe DEPendeNce measure
IMGMSE         IMaGe Mean-Squared Error & signal-to-noise comparison
IMGDUMP        IMaGe DUMP to user_out
COPYIM         COPY IMage file to image file
RDHEADER       READ image file HEADER string
WRHEADER       WRite image file HEADER string
```

```
*
*   MODEL 70 IMAGE FUNCTIONS
*
*
MWINMASK       M70 WINdow MASKing general convolution operator
MSOBEL         M70 SOBEL gradient operator
MNOISECLN      M70 NOISE CLeaNing operation
MROBERTS       M70 ROBERTS gradient operator
MAVERAGE       M70 window AVERAGE operator
FMAVERAGE      Fast M70 window AVERAGE operator
MLAPLACN       M70 LAPLACiaN operator
MSCALEACC      M70 SCALE ACCumulator
MCOMPASS       M70 COMPASS gradient operator
MABSACC        M70 ABSolute value ACCumulator
MACCDIV        M70 DIVide ACCumulator by constant
MHISTHYP       M70 HISTogram HYPerbolization
MHISTEQU       M70 HISTogram EQUalization
MWINMAX        M70 WINdow MAXimum
MWINMIN        M70 WINdow MINimum
M7MEDIAL       M70 MEDIAL axis operator
M72FFT         M70 2 Dimensional Fast Fourier Transform
M72FHT         M70 2 Dimensional Fast Hadamard Transform
*
*   MODEL 70 SUPPORT ROUTINES
*
*
M70_INIT       M70 INITialize
M7CLEAR        M70 CLEAR channels
MPLOT          M70 PLOTting
M7CONN         M70 channel CONNECTion
M7COPY         M70 channel COPY
M7STATE        M70 STATE checker
PSEUDOCOLOR    PSEUDO COLOR generator
TBASCROLL      TrackBall Annotated SCROLL
```

```
*
*   FEATURE VECTOR FUNCTION COMMANDS
*
*
--none yet--
*
*   FEATURE VECTOR SUPPORT COMMANDS
*
*
CREATFV         CREATe Feature Vector file
FVPRINT         Feature Vector file PRINT
FVCOPY          Feature Vector file COPY
FVINFO          Feature Vector file INFOrmation
FVNORMAL        Feature Vector file NORMALization
FVCHANGE        Feature Vector file CHANGEing
CLASSINFO       CLASS INFOrmation
```

```
*
*   COMMAND SUBSYSTEMS
*
*
COM70          COMmand processor for m70 (see manual)
*
*   MISCELLANEOUS COMMANDS
*
*
SEECOM         SEE COMmand (for debugging abbrevs)
INFO           get INFOrmation on subroutine or command
LROFF          text formatter (see manual)
DELAY          DELAY ec file execution
PAUSE          PAUSE ec file execution
TAPDSK         copy foreign TAPes to DiSK
LAS            List ASsociations
AVLMEM         return AVaiLable MEMory size
PRL            PRint with Line numbers
RDMEMO         Read MEMO file
WRMEMO         WRite MEMO file
PF             Print File with pauses every 23 lines
C1DFFT         Cpu 1 Dimensional Fast Fourier Transform
*
*   ACTIVE FUNCTIONS
*
*
TBSUB          TrackBall SUBimage definition (one relative)
TBPOS          TrackBall POSition (one relative)
CURSUB         CURsor SUBimage definition (zero relative)
CURPOS         CURsor POSition (zero relative)
EXPAND         EXPAND relative pathname
INFO           return pathname of installed software
FVPRINT        Feature Vector PRINTing
MAX            find MAX number in a real list or file
MIN            find MIN number in a real list of file
*
*   EC FILES  (POINTER TO EC FILES IN IRLAB.AB)
*
*
LIST           print (LIST) all files in directory using star specifier
SEARCH         SEARCH all files in directory for character string
EBC            Enter Batch Command
PHONE          search PHONE list
START_UP       example START_UP ec file
LINK           example LINKing ec file
CHAINSUB       example linking ec file for CHAINing SUBroutines
CHAINCOM       example linking ec file for CHAINing COMmands
```